Latvijas Universitāte
Datorikas fakultāte

CPU. DataPath. ALU

Kurss "ievads digitālajā projektēšanā"
Lekcija 28.09.2012

Autors: Rinalds Ruskuls

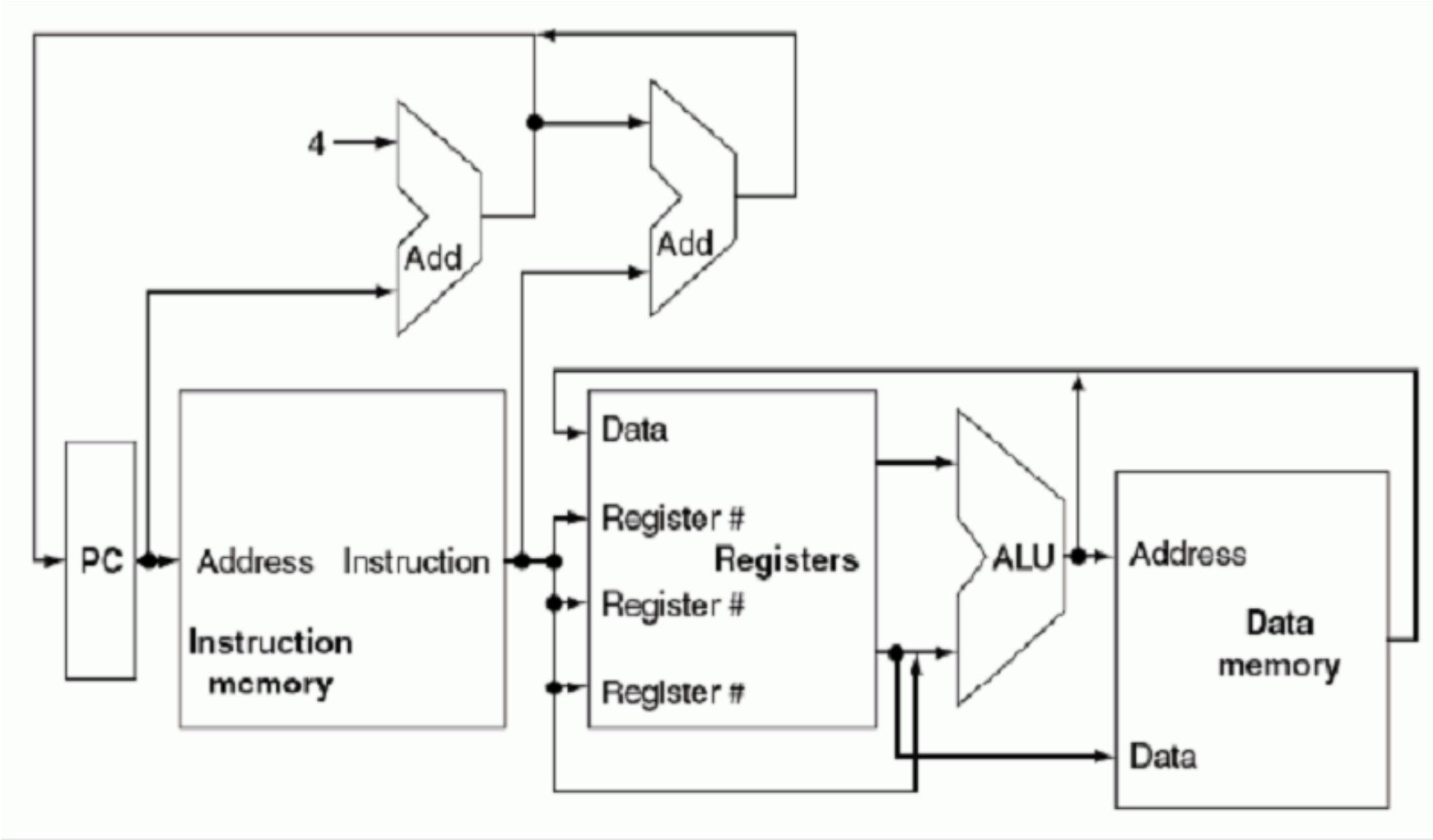
Lekcijas saturs

- MIPS arhitektūras CPU
- ALU
  - Loģiskās darbības
  - Aritmētiskās darbības
  - Salīdzināšanas darbības

MIPS

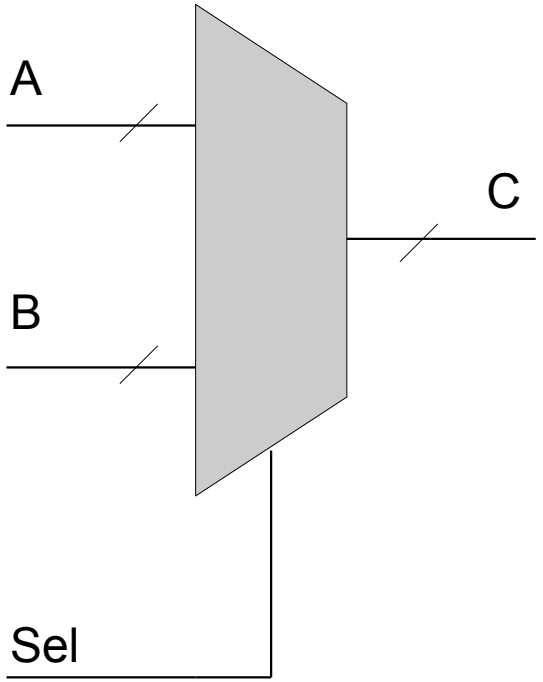
- **M**icroprocessor without **I**nterclocked **P**ipeline **S**tages
- Pieder pie RISC (**R**educed **I**nstruction **S**et **C**omputing)
- Eksistē 32 un 64 bitu versijas (šajā kursā runāsim par 32 bitu versiju)
- Šāda tipa mikroprocesori tiek izmantoti rūteros (Cisco), spēļu konsolēs (PlayStation 1&2), kā arī citās iegultās sistēmās

MIPS procesors (vienkāršots)



Multiplekseri (MUX)

- Reizēm mums vienā ieejā ir jāpadod vairāki signāli, turklāt katrs savā laikā

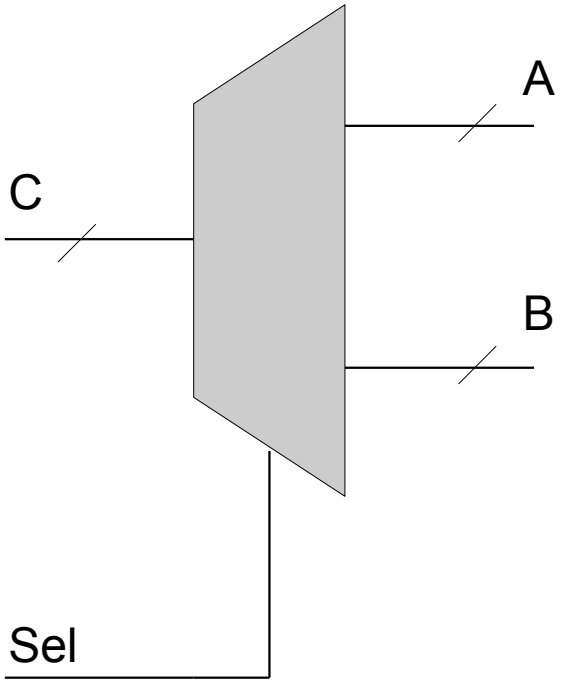


Sel	C
0	A
1	B

Kā aprēķināt, cik bitu Sel vērtība jāizvēlas no ieejas signāliem?

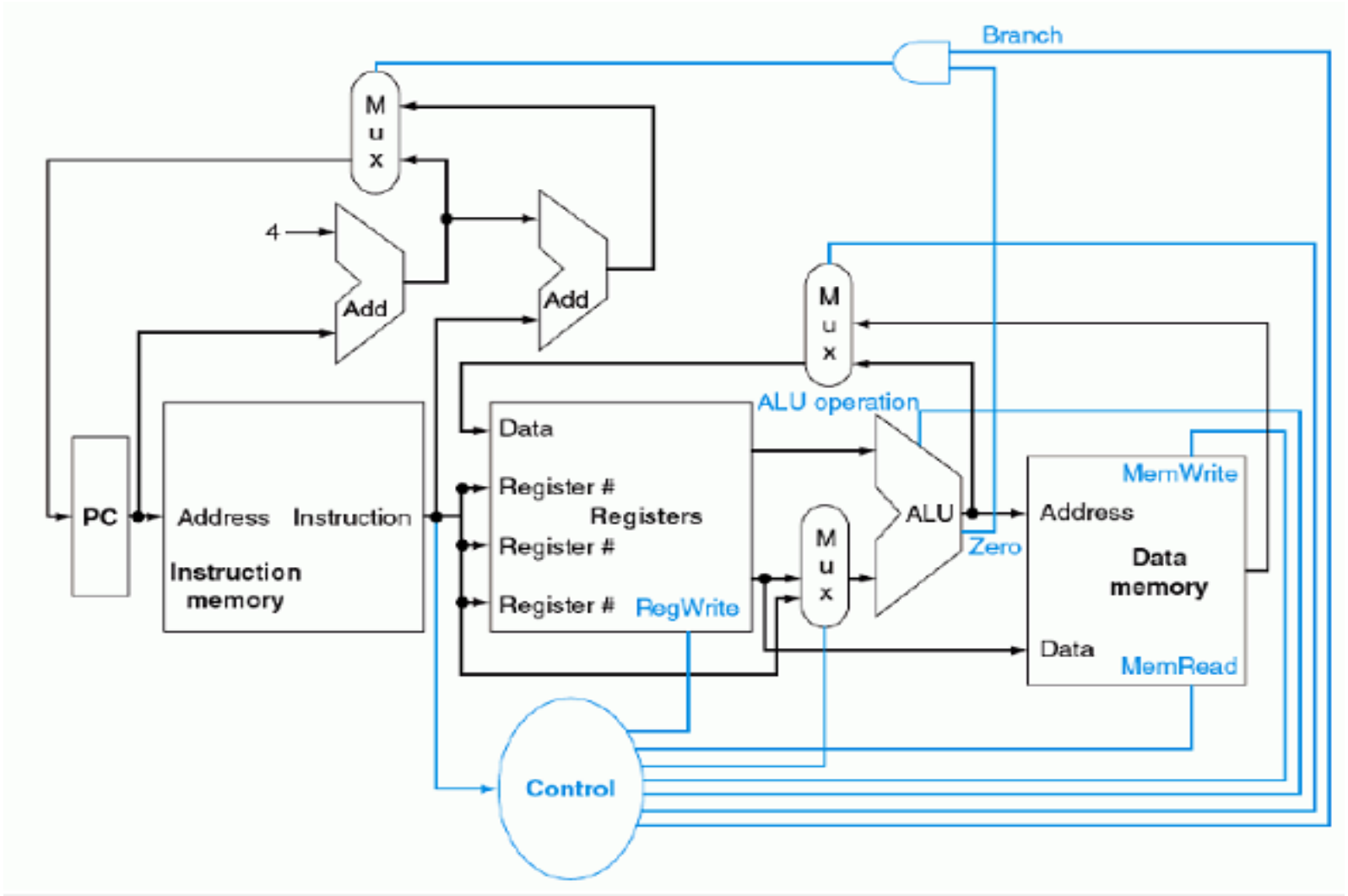
DeMultiplekseri (DeMUX)

- Pretēja darbība multipleksoram – vienu ieejas signalu piešķiram vairākiem izejas signāliem



Sel	A	B
0	C	0
1	0	C

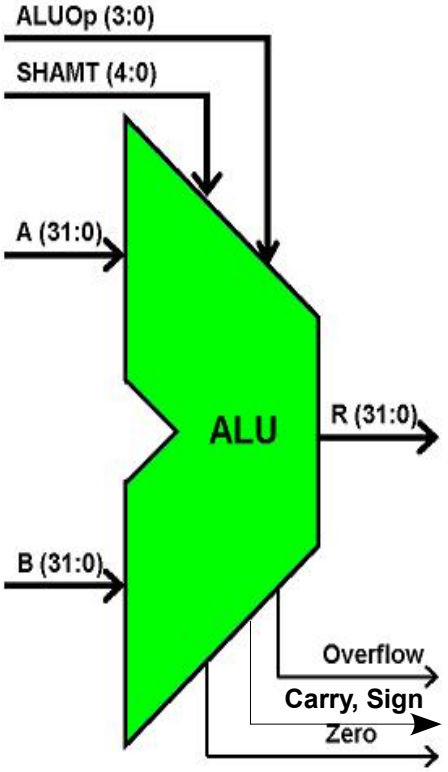
MIPS procesors (detalizētāks)



Aritmētiskais-loģiskais mezgls (**A**rithmetic-**L**ogic **U**nit)

ALU pamatoperācijas

- Loģiskās (AND, OR, XOR, NOT)
- Aritmētiskās (“+”, “-”, “·”, “/”)
- Bitu nobīdes operācijas
- Salīdzināšanas operācijas.



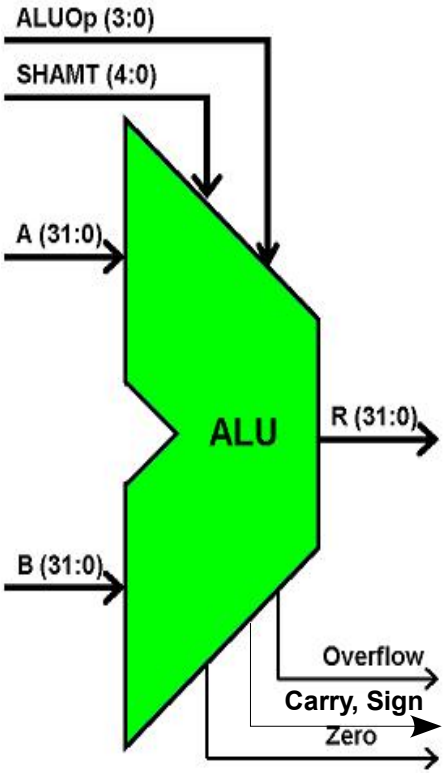
Aritmētiskais-loģiskais mezgls (**A**rithmetic-**L**ogic **U**nit)

Ieejas

- A, B – operandi, ar kurām veic operācijas
- ALUOp – izpildamās operācijas kods
- SHAMT – operandu bitu nobīdes skaits

Izejas

- R – operācijas rezultāts
- Flags (“karodziņi”)
  - Overflow – pārpildes indikācija
  - Zero – nulles rezultāta indikācija
  - Carry – pārnese uz 32. kārtu
  - Sign – negatīvā rezultāta indikācija, =R[31]



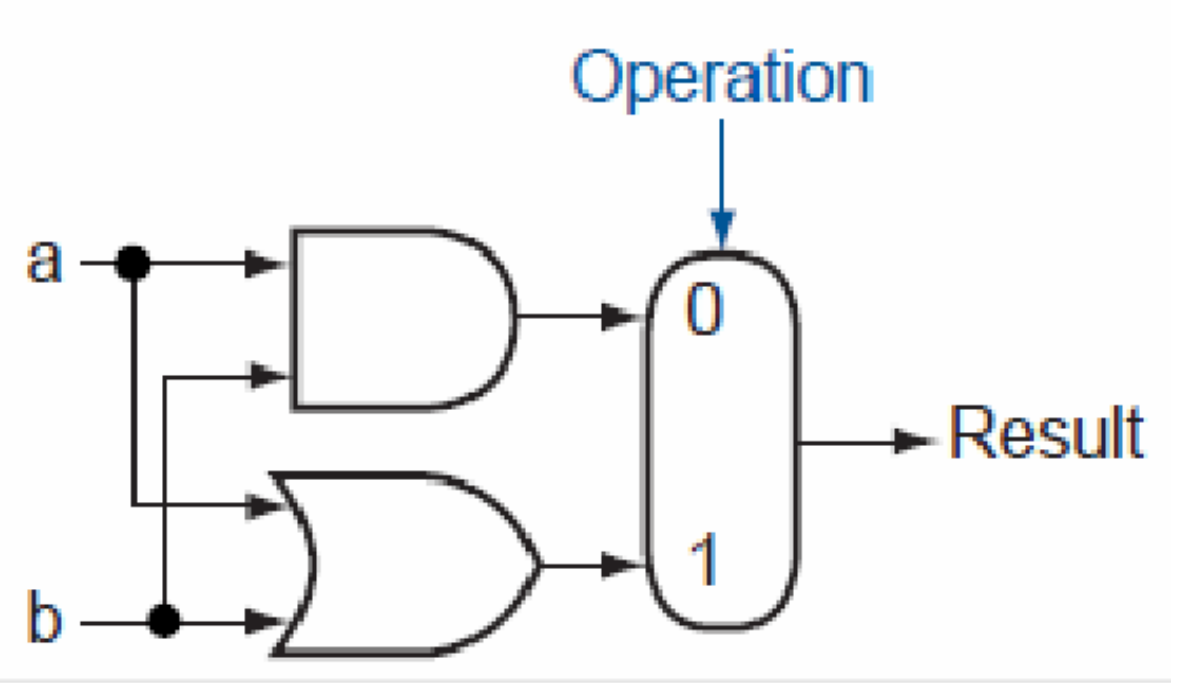
Aritmētiskais-loģiskais mezgls (**A**rithmetic-**L**ogic **U**nit)

ALU operāciju piemēri:

ALU operācija	Apraksts
Addition (ADD)	R = A + B
Subtraction (SUB)	R = A - B
Bitwise NOT	R[i] = NOT A[i]
Bitwise AND	R[i] = A[i] AND B[i]
Bitwise OR	R[i] = A[i] OR B[i]
Bitwise XOR	R[i] = A[i] XOR B[i]
Shift Left Logical (SLL)	R = A << SHAMT; R[0...(SHAMT-1)] = 0
Shift Right Logical (SRL)	R = A >> SHAMT; R[(32-SHAMT)...31] = 0
Shift Right Arithmetic (SRA)	R = A >> SHAMT; R[(32-SHAMT)...31] = A[31]
...	

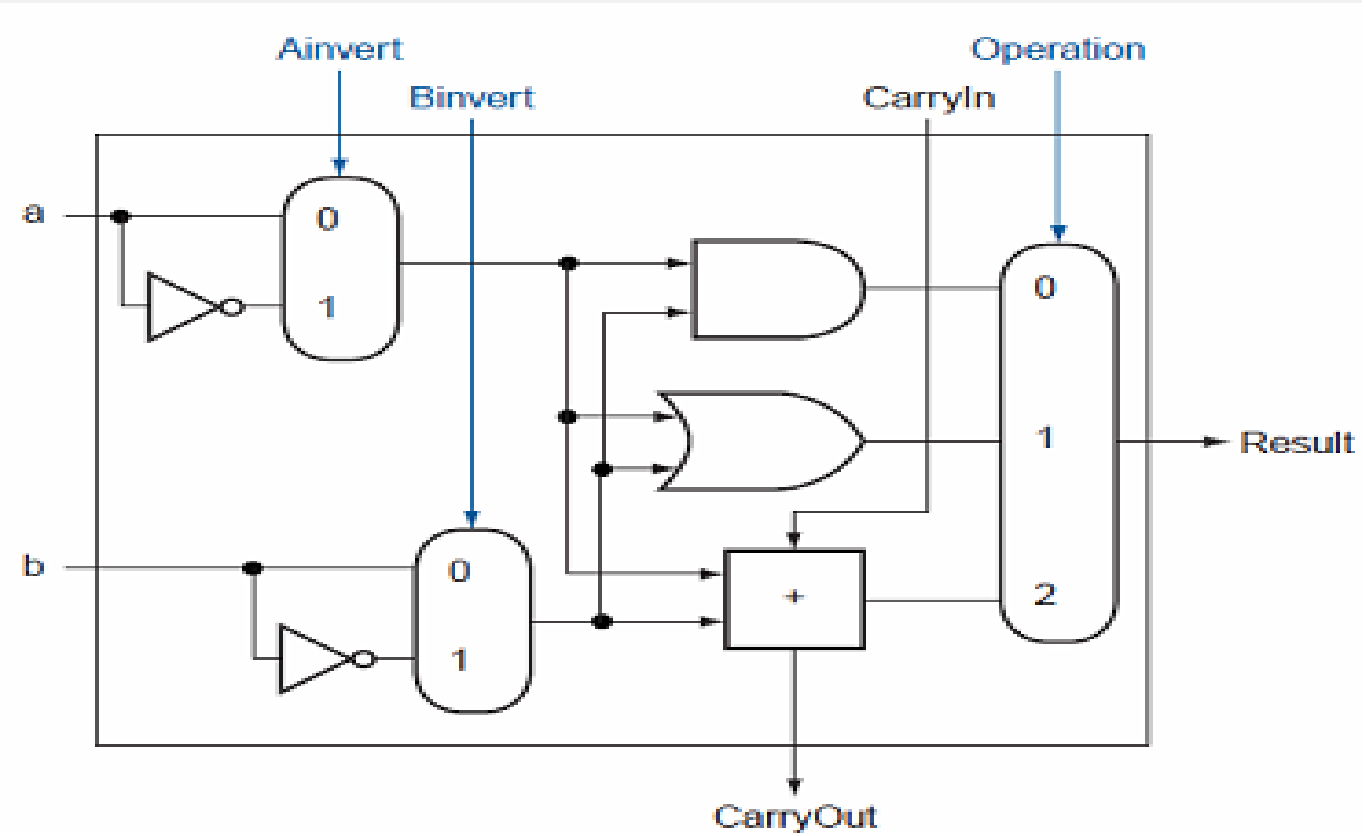
Loģiskās funkcijas – AND, OR

- Loģisko funkciju izvēlas ar "Operation" mainīgo



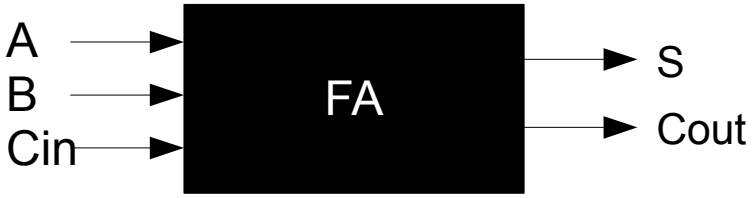
Loģiskā operācija NOR

- Vai varam izmantot jau esošās loģiskās funkcijas?
- NOT (A OR B) = NOT(A) AND NOT(B)



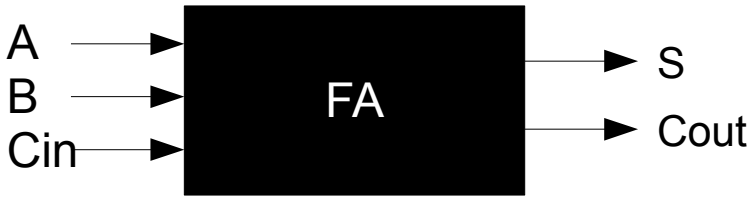
Aritmētiskās operācijas

1 bita saskaitītājs (Full Adder)

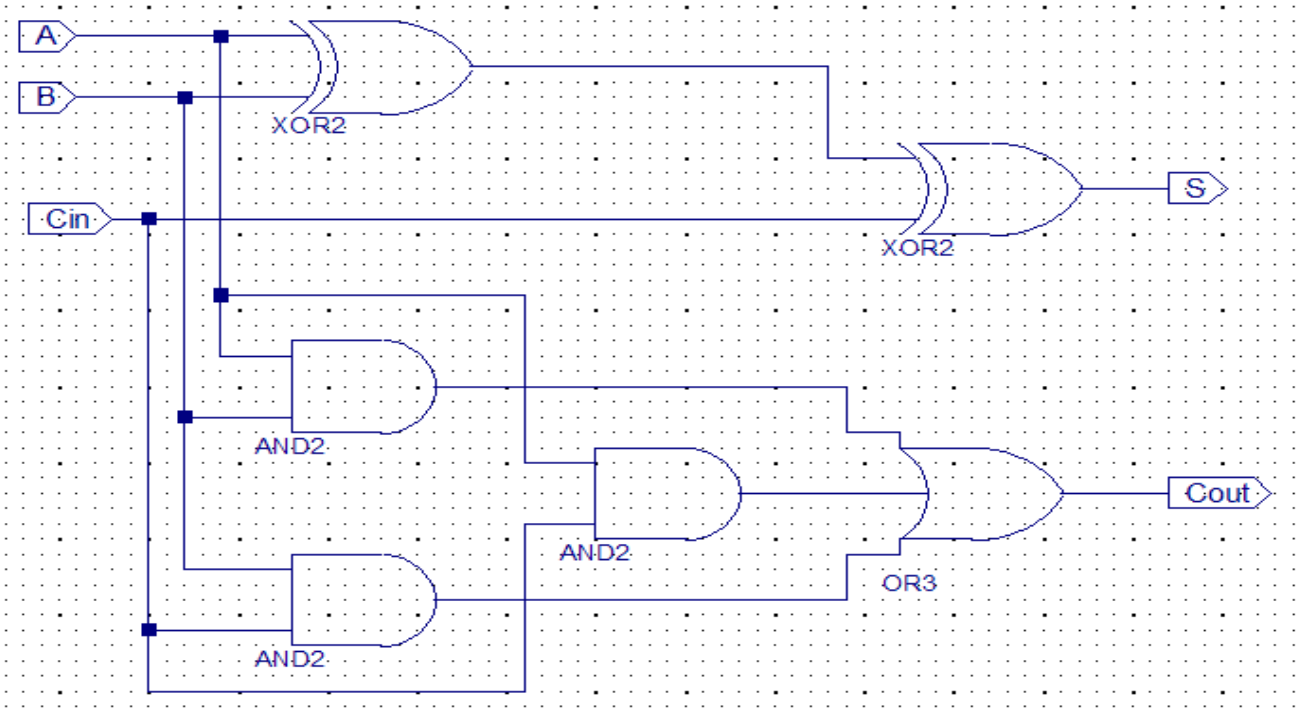


A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

1 bita saskaitītājs (Full Adder)



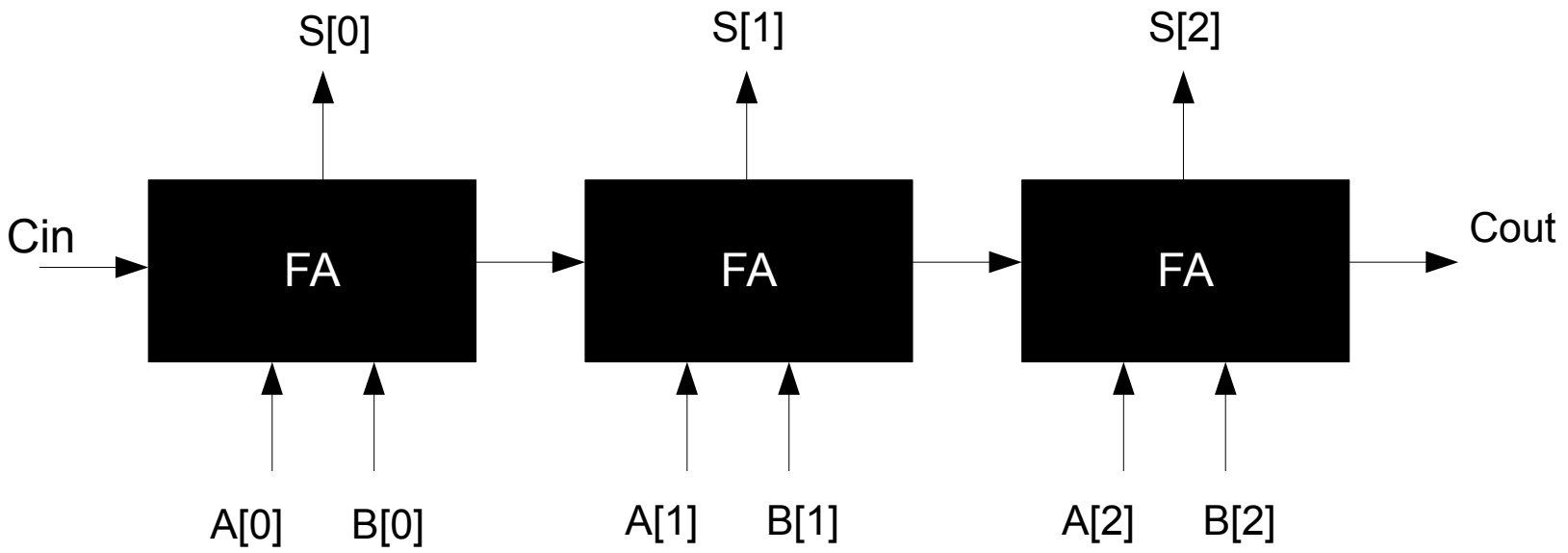
A	B	Cin	Cout	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



3 kārtu saskaitītājs (Full Adder)

- Liekot kaskādē šādus saskaitītājus var iegūt vairāku bitu summatorus

- Problēmas šajā risinājumā?



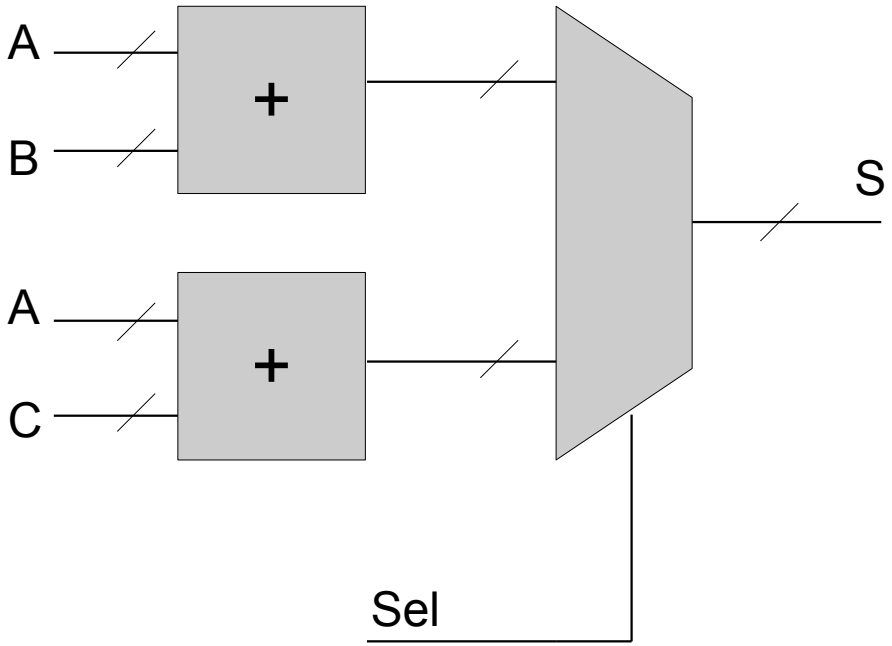
Uzdevums

- Izveidot shēmu, kas spēj saskaitīt
  - A + B, ja Sel = 0;
  - A + C, ja Sel = 1;

- Jūsu piedāvātie risinājumi?

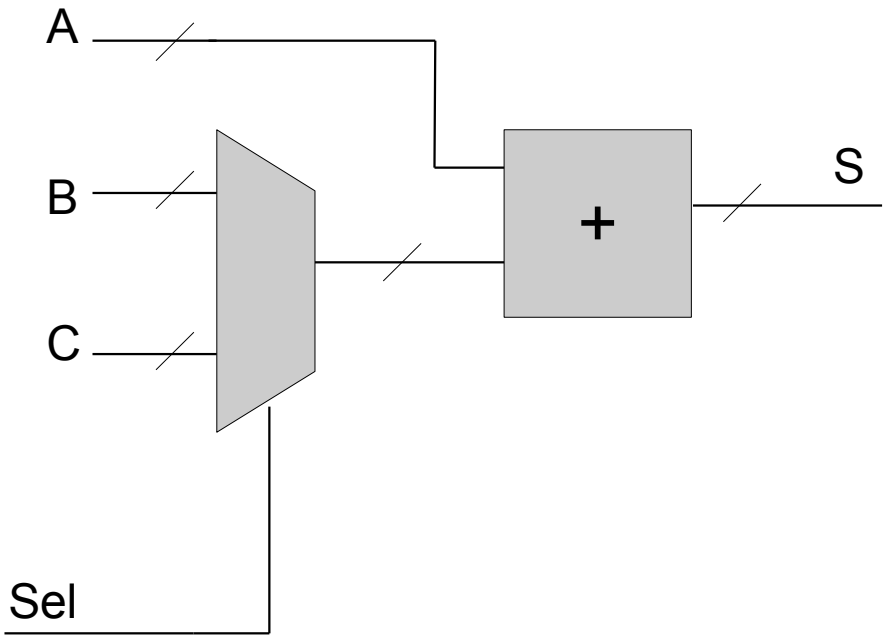
Risinājums #1

- Izmanot divus summatorus un vienu MUX, cik tas ir efektīvi?



Risinājums #2

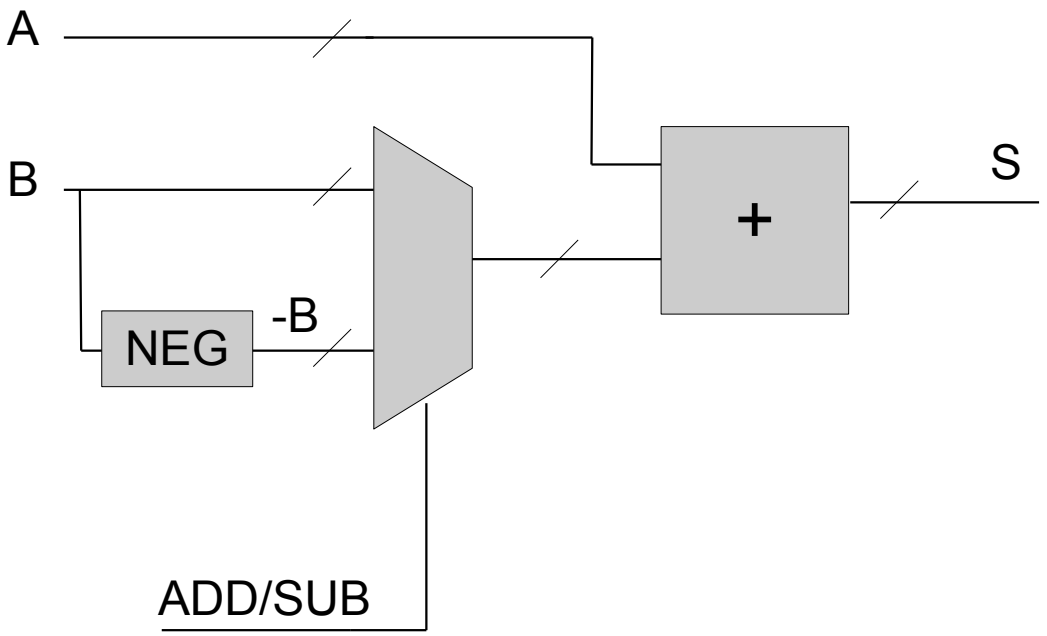
- Var izmantot vienu multiplekseri un vienu summatoru



Kā implementēt atņemšanu?

Atņemšanas realizācija

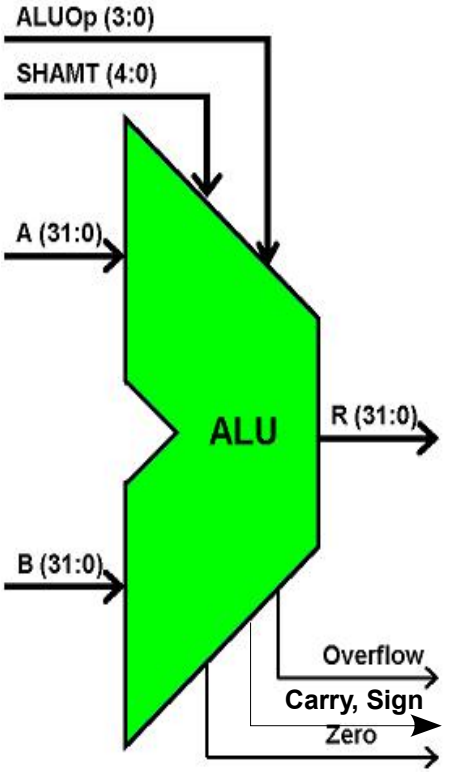
- Izmanto papildkodus
- Kas būtu jādara, lai pievienotu *dec* un *inc* funkcionalitāti?



Skaitļu salīdzināšana

Salīdzinot skaitļus izpilda sekojošas operācijas

1. R = A – B
2. Iegūst Flags vērtības
   - A=B, ja Zero=1, jo tad R=A–B=0
   - A<B, ja Zero=0, Sign=1, jo tad R=A–B<0
   - A>B, ja Zero=0, Sign=0, jo tad R=A–B>0



Mājas darbs #3

- Iesūtīšanas termiņš – 5.10.2012

- Izveidot loģisko shēmu, kas nodrošina sekojošu funkcionalitāti
  - Ieejas – A,B (8 biti)
  - Izejas – Q (8 biti), Carry out, overflow
  - Q = A + B
  - Darbojas ātrāk nekā lekcijā aprakstītais risinājums

Paldies par uzmanību!
Jautājumi?

Pārpilde - overflow

- Pārnese – bezzīm.es vērtību saskaitīšana vai atņemšana, kas neietilpst rezultātam atvēlētajā vietā. Lai to konstatētu tiek pārbaudīts pārneses korodziņš

```
    1111 (4 biti unsigned 15)
+   1111 (4 biti unsigned 15)
---------------------------------
  1 1110 (4 biti bez pārneses 14, vai 5 biti ar pārnesi 30)
```

- Pārpilde – aritmētisku operāciju rezultātam ir zime, kura ir pretrunā ar operandu zīmēm, piemēram, divu pozitīvu saskaitāmo summa ir negatīva. Lai to konstatētu tiek pārbaudīts pārpildes karodziņš

```
    01111111 (8 biti signed 127)
+   01111111 (8 biti signed 127)
---------------------------------
    11111110 (8 biti bez pārpildes - 2, vai 9 biti ar pārpildi 127)
```

Datapath

- Procesora daļa, kas veic datu apstrādes operācijas
- Pie *datapath* **nepieder** kontroles bloks